Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Peter Jurše

Avtomatsko konstruiranje z orodjem SolidWorks

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: izr. prof. dr. Uroš Lotrič

Ljubljana 2014

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Proizvodna podjetja se danes v borbi za kupce trudijo, da bi izdelke čim bolj prilagodila njihovim željam in potrebam. Zaradi velike raznolikosti je proizvodni proces dolgotrajen in neučinkovit. Kljub uporabi podpornih sistemov je ročno načrtovanje vsakega izdelka posebej zelo zamudno in ne omogoča hitrega odziva podjetja na želje kupca. V diplomskem delu izdelajte programski paket, ki bo za izbrane izdelke omogočal avtomatsko pripravo strojniških risb in kosovnic ter tako pripomogel k izboljšanju kakovosti proizvodnega procesa.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Peter Jurše, z vpisno številko **63090002**, sem avtor diplomskega dela z naslovom:

Avtomatsko konstruiranje z orodjem SolidWorks

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom izr. prof. dr. Uroša Lotriča,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 8. septembra 2014 Podpis avtorja:

Mentorju izr. prof. dr. Urošu Lotriču se iskreno zahvaljujem za pomoč pri izdelavi diplomskega dela. Posebej bi se rad zahvalil vsem v podjetju IB-CADDY d.o.o., ki so mi pri delu svetovali, ter tudi družini, ki me je pri tem podpirala.

Kazalo

Kazalo slik

Kazalo izsekov kode

Seznam uporabljenih kratic

kratica	angleško	slovensko
CAD	Computer-aided Design	računalniško podprto načrtovanje
API	Application Programming Interface	programski vmesnik
SDK	Software Development Kit	orodje za razvoj programske opreme
IDE	Integrated Development Environment	integrirano razvojno okolje
CSV	Comma-Separated Values	podatki, ločeni z vejico
COM	Component Object Model	komponentni objektni model

Povzetek

Cilj diplomske naloge je bil avtomatizirati konstrukcijo proizvodnega izdelka v sistemu SolidWorks v povezavi s programom ciciVO, ki služi kot podporno orodje pri prodaji vgradnih omar. Izdelali smo dodatek za SolidWorks, ki iz projektne datoteke omenjenega programa pridobi informacije o določenem izdelku in s pomočjo teh zgradili njegov 3D-model v sistemu SolidWorks. V ta namen je bilo treba izdelati uporabniški vmesnik dodatka in napisati algoritem za gradnjo. Pri tem smo se morali sprehoditi skozi drevesno strukturo, opisano v projektni datoteki, in zgraditi enako strukturo v SolidWorksu. Na koncu je tako potreben le pritisk na gumb in v nekaj minutah je 3D-model pripravljen za proizvodnjo. Na ta način smo bistveno skrajšali proces od prodaje do proizvodnje naročenega izdelka, saj je bilo prej za enak, vendar ročni postopek potrebnih več ur dela.

Ključne besede: avtomatizacija, CAD, SolidWorks, proizvodnja, izdelek, vgradna omara.

Abstract

The goal of this thesis was to automate the construction of manufacturing products in SolidWorks in connection with the ciciVO software that serves as a support tool in the sales of built-in closets. We have developed the SolidWorks add-in that obtains information about a particular product from the project file of the previously mentioned software and uses it to build its 3D model in SolidWorks. For this purpose we had to create a user interface for the add-in and programme the construction algorithm. This included going through the tree structure described in the project file and building the same structure in SolidWorks. In the end just a press of the button is required and in a few minutes the 3D model is ready for production. In this way, we have significantly shortened the process from the sale of the ordered product to production, as the similar manual process previously required several hours of work.

Keywords: automation, CAD, SolidWorks, production, product, built-in closet.

Poglavje 1

Uvod

Načrtovanje izdelka je za proizvodnjo ključnega pomena, saj lahko slabo načrtovanje privede do dodatnih stroškov in zamud v proizvodnem procesu, kar neposredno vpliva na uspeh podjetja oziroma njegov položaj na trgu. Zato večina podjetij v ta namen uporablja sisteme za računalniško podprto načrtovanje (angl. Computer-aided design – CAD), ki veliko prispevajo k poenostavitvi in optimizaciji načrtovanja. Z njihovo uporabo se bistveno zmanjšata tako čas proizvodnje kot količina uporabljenega materiala, razen tega pa se praktično izniči možnost napak v proizvodnem procesu. Kljub uporabi programske opreme CAD pa se srečamo s težavami pri načrtovanju, ko ne gre za izdelek, namenjen množični proizvodnji, in ne zadošča enkraten izris 3D-modela. Eden izmed takšnih izdelkov je na primer vgradna omara, saj omogoča veliko število različnih konfiguracij. Že njena velikost je po navadi odvisna od prostora, v katerega bo postavljena, še več možnih variacij pa nastane na račun preferenc glede notranjosti, kjer si lahko naročnik po želji zamisli postavitev polic, predalov, obešalnikov itn.

Z vnovičnim načrtovanjem za vsak primerek izdelka bi izgubili veliko dragocenega časa, to pa je tudi problem, ki ga želimo premostiti v okviru te diplomske naloge. Glavni cilj predstavlja razvoj dodatka za SolidWorks (angl. Add-In), ki bi poskrbel za avtomatizacijo gradnje 3D-modela in s tem za izrazito skrajšanje proizvodnega časa takega izdelka. Njegova naloga bo, da izhodno datoteko prodajnega programa ciciVO uporabi za generiranje 3D-modela v sistemu SolidWorks. Pri tem naj uporabi vnaprej pripravljeno knjižnico posameznih v SolidWorksu izrisanih komponent in datoteko CSV, ki bo služila kot preslikava med tipi komponent

in njihovimi lastnostmi, kot sta ime in tip datoteke v prej omenjeni knjižnici, ter druge potrebne informacije. Po uspešno zaključeni gradnji bomo dodali še kosovnico, ki bo vsebovala število posameznih komponent in njihove dimenzije, s čimer bomo še dodatno poenostavili delo v proizvodnji. V našem primeru se bomo sicer soočili s problemom vgradne omare, vendar pa želimo algoritem zastaviti tako, da bo z ustrezno zamenjavo vhodnih podatkov mogoče avtomatizirati tudi gradnjo katerega koli drugega izdelka in ga tako pripraviti za proizvodnjo oziroma nadaljnjo uporabo v sistemih CAD.

Sistem CAD predstavlja kombinacijo strojne in programske opreme, ki omogoča izdelavo dvo- ali tridimenzionalnih grafičnih upodobitev različnih objektov. Do sredine osemdesetih let so vse sisteme CAD sestavljali namensko izdelani računalniki, zdaj pa lahko programska oprema CAD teče na splošnih delovnih postajah in osebnih računalnikih. Veliko se uporablja v avtomobilski, ladijski in letalski industriji, pri industrijskem in arhitekturnem načrtovanju ter protetiki, pa tudi za računalniške animacije in posebne učinke v filmih ter oglaševanju. Programska oprema za računalniško podprto načrtovanje zajema vse od 2D-sistemov za vektorsko risanje pa do 3D-modelirnikov teles in površin. Ločimo torej 2D- in 3D-CAD. Prvi se pogosteje uporablja na primer v arhitekturi za oblikovanje načrtov, kot so tlorisni pogledi stavb, medtem ko je 3D-CAD med drugim prisoten pri načrtovanju proizvodnih izdelkov in razvoju videoiger. Tako 2D- kot 3D-CAD navadno temelji na vektorski grafiki. To omogoča oblikovalcem, da ustvarjajo kompleksne (predvsem 3D) oblike, ki jih je mogoče premikati, vrteti in jim spreminjati velikost. Prav tako jih je mogoče brez izgube na kakovosti približati na poljubno oddaljenost in na ta način ustvariti tako podrobne načrte kot tudi preglede širšega prostora. Številni programi CAD oblikovalcu omogočajo barvanje površin in uporabo tekstur ter možnost prilagajanja osvetlitve, ki vpliva na sence in odseve modela. Nekateri programi vključujejo tudi časovnico, ki omogoča ustvarjanje 3D-animacij. Zaradi razširjenosti računalniško podprtega načrtovanja je tudi programska oprema pogosto specializirana in prilagojena posameznemu področju uporabe, saj so določene funkcije za neko področje lahko nepomembne in celo neuporabne, s svojo prisotnostjo pa le škodijo preglednosti in enostavnosti programa za uporabo. [1]

CAD je danes ena izmed najpomembnejših računalniško podprtih tehnologij (angl. computer-aided technologies), kamor med drugim spadajo tudi računalni-

ško podprta proizvodnja (angl. computer-aided manufacturing), avtomatizacija načrtovanja električnih vezij (angl. electronic design automation) in na primer računalniška dinamika tekočin (angl. computational fluid dynamics). [2]

Čeprav se v industrijskem načrtovanju CAD uporablja predvsem za izdelavo podrobnih 3D-modelov ali 2D-risb posameznih fizičnih komponent, je njegova uporaba prisotna skozi celoten inženirski proces – vse od idejne zasnove izdelka, skozi analize trdnosti in dinamike posameznih komponent, do izbire proizvodne tehnologije. To inženirjem omogoča tako interaktivno kot tudi samodejno analizo različnih variant izdelka, s čimer lahko poiščejo optimalen dizajn za proizvodnjo in hkrati tudi minimizirajo uporabo fizičnih prototipov.

Uporabe programske opreme CAD nudi številne prednosti:

- boljša vizualizacija končnega izdelka in njegovih sestavnih delov v sistemu CAD pospeši proces načrtovanja;
- programska oprema CAD omogoča večjo natančnost, kar pripomore k zmanjšanju napak;
- sistem CAD nudi enostavnejšo in robustnejšo pripravo dokumentacije, ki med drugim vključuje dimenzije in kosovnice;
- programska oprema CAD omogoča preprosto ponovno uporabo obstoječih dizajnov in znanih najboljših praks.

Vse naštete prednosti prispevajo k nižjim stroškom proizvodnje, povečanju produktivnosti in kakovosti izdelka ter h krajšemu času, ki je potreben do začetka trženja izdelka. [3]

V naslednjem poglavju bomo pregledali orodja in tehnologije, ki nam bodo v pomoč pri delu. Ogledali si bomo pomembnejše funkcionalnosti enega izmed sistemov CAD, imenovanega SolidWorks, ki ga bomo kasneje vključili tudi v praktični del diplomske naloge, kjer bomo končno razvili dodatek za SolidWorks.

Poglavje 2

Uporabljena orodja in tehnologije

Preden se posvetimo razvoju programske rešitve za naš problem, preglejmo še nekatera orodja ter tehnologije, ki jih bomo pri tem uporabili. Brez nekaterih izvedba sploh ne bi bila mogoča, druga pa nam le-to poenostavijo oziroma nam omogočajo, da se bolj osredotočimo na sam razvoj.

2.1 Visual Studio

Pri implementaciji dodatka za SolidWorks nam bo v pomoč Microsoft Visual Studio, ki je integrirano razvojno okolje (angl. Integrated development environment – IDE) in se uporablja za razvoj aplikacij, namenjenih operacijskemu sistemu Windows, kot tudi spletnih strani ter spletnih aplikacij in storitev. Pri tem so nam na voljo napreden urejevalnik programske kode, prevajalnik, povezovalnik, razhroščevalnik, orodja za idelavo grafičnega uporabniškega vmesnika, omogoča pa tudi razširitev funkcionalnosti z namestitvijo številnih vtičnikov (angl. Plug-Ins). Jezikovni servis (angl. Language service) ponuja možnost namestitve podpore za programiranje v praktično vseh programskih jezikih. V osnovi pa so na voljo jeziki C, C++, VB.NET, C# in F#.

2.2 Programski jezik C#

Za razvoj našega dodatka bomo uporabili programski jezik C#. C# je močno tipiziran programski jezik, ki nudi podporo več programerskim paradigmam, kot so imperativno, deklarativno, funkcijsko, generično in objektno orientirano programiranje. Prvo različico jezika je Microsoft v okviru ogrodja .NET razvil že leta 1999, trenutno pa je aktualna izdaja 5.0, ki je izšla avgusta 2012.

2.3 Zapis CSV

Zapis CSV predstavlja preprost način shranjevanja in organiziranja podatkov. Čeprav format datoteke CSV ni standardiziran, obstaja nekaj pravil, ki definirajo, kako naj bi ga uporabljali:

- CSV je podatkovni format, ki vsebuje polja/stolpce, med seboj ločene z vejico, in zapise/vrstice, ki so med seboj ločeni s prelomom vrstice;
- datoteka CSV ne zahteva posebnega kodiranja, vrstnega reda zapisovanja bajtov ali specifičnega zaključevanja vrstic;
- prelom vrstice običajno predstavlja konec zapisa. Redko je podprt prelom vrstice kot del podatkovnega polja. Takrat mora biti podatkovno polje obdano z navednicami;
- vsi zapisi morajo vsebovati enako število podatkovnih polj, ki morajo biti tudi v enakem zaporedju;
- podatkovna polja so interpretirana kot zaporedja znakov in ne kot zaporedja bitov ali bajtov. Na primer količina, predstavljena z neko številko, recimo 12345, je zapisana kot zaporedje petih ASCII-znakov: »12345«. Če se ne bi držali tega dogovora, datoteka CSV ne bi več vsebovala zadostne informacije za pravilno interpretacijo;
- presledki so tudi interpretirani kot del podatkovnega polja;
- podatkovna polja lahko obdamo tudi z navednicami;
- podatkovna polja, ki vsebujejo vejico, morajo biti obdana z navednicami;

- zaporedna polja morajo biti ločena z eno vejico. V primeru dveh zaporednih vejic je vmes predstavljeno prazno polje. Sicer lahko polja loči tudi kakšen drug dogovorjeni znak, predvsem na področjih, kjer vejica predstavlja decimalni separator. Takrat lahko uporabimo podpičje, tabulator, kakšen drug znak ali zaporedje znakov;
- prvi vnos je pogosto »glava«, ki v podatkovnih poljih vsebuje imena stolpcev.

Format CSV se pogosto uporablja za shranjevanje podatkov različnih preglednic, zaporedij zapisov z identičnim zaporedjem podatkovnih polj in tudi podatkovnih baz. Zaradi preprostosti ga bomo tudi mi uporabili za hranjenje nekaterih vhodnih podatkov. Ker je tudi enostavno razširiti podatkovno strukturo, ki jo opisuje, nam bo to prišlo prav pri morebitnih razširitvah delovanja, ki bi zahtevale dodatne vhodne podatke. [4]

2.4 SolidWorks

Eden izmed prej omenjenih sistemov CAD je programski paket SolidWorks, ki je zelo priljubljen zaradi enostavne uporabe, širokega spektra funkcij, ki jih ponuja, in zaradi nižjih stroškov uporabe v primerjavi s konkurenčnimi rešitvami. Tudi v tej diplomski nalogi ima pomembno vlogo, saj na njem temelji delovanje naše programske rešitve in na ta način predstavlja nekakšno jedro oziroma ogrodje.

SolidWorks je produkt podjetja Dassault Systèmes SolidWorks Corp., ki je hčerinsko podjetje Dassault Systèmes, S. A. s sedežem v Franciji. Korporacijo SolidWorks je decembra 1993 ustanovil Jon Hirschtick, diplomant na Massachusetts Institute of Technology (MIT), leta 1997 pa je lastništvo prevzelo podjetje Dassault. Po podatkih iz leta 2013 je v drugem četrtletju SolidWorks uporabljalo več kot 2 milijona inženirjev in oblikovalcev v več kot 180.000 podjetjih. Do danes je bilo po celem svetu prodanih že več kot dva milijona licenc, pri čemer tričetrtinski delež zajemajo licence, izdane v namen izobraževanja. [5] Programsko opremo SolidWorks uporabljajo tako posamezniki kot tudi večje korporacije, s čimer zajema zelo širok del proizvodnega tržnega segmenta. V podjetju sodelujejo tudi z zunanjimi razvijalci programske opreme z namenom dodajanja funkcionalnosti za nišne tržne segmente, kot so analiza končnih elementov, načrtovanje vezij, preverjanje toleranc itd.

2.4.1 Delovanje

SolidWorks temelji na modelirnem jedru Parasolid, ki je v lasti podjetja Siemens PLM Software. Za izdelavo modelov in sestavov uporablja parametrični pristop na osnovi funkcij. Parametri modela so vezani na omejitvene pogoje, katerih vrednosti določajo obliko geometrije modela. Parametri so lahko številski, na primer dolžina črte in premer kroga, ali pa so geometrijski, kot so vzporednost, koncentričnost, vodoravnost, navpičnost itd. Parametri so lahko tudi v relaciji med seboj in na ta način opisujejo želeni dizajn. Preprosteje povedano, če oblikujemo model pločevinke za pijačo, lahko določimo, da bo odprtina vedno na zgornji ploskvi, ne glede na višino modela. Tudi če kasneje spremenimo parameter, ki določa višino, bo še vedno veljalo omenjeno pravilo. Končno podobo 3D-modela določa množica funkcij (angl. Features). Med njimi so lahko razne oblike in operacije, ki z uporabo ene za drugo vodijo do želenega rezultata. Načrtovanje se po navadi začne z 2D- ali 3D-skico, ki vsebuje geometrijske elemente, kot so točke, črte, loki, stožnice in zlepki. Skici nato dodamo novo prostorsko dimenzijo, da določimo velikost in položaj geometrije. Zatem lahko uporabimo tudi že prej omenjene relacije med parametri, če želimo na primer doseči vzporednost dveh ploskev. Parametrična narava SolidWorksa pomeni, da dimenzije in relacije določajo geometrijo, in ne obratno. Dimenzije na skici lahko nadzorujemo neodvisno ali pa v relaciji z drugimi parametri znotraj ali izven skice. Več o parametričnem in funkcijskem konceptu CAD si lahko preberemo v [6] in [7].

Kadar imamo 3D-model, ki vsebuje več kosov (angl. Parts), mu rečemo sestav (angl. Assembly). Tudi med kosi lahko uporabimo relacije, da dosežemo pravilno pozicioniranje in morebitno gibanje pri kosih, ki niso fiksni. Po končanem načrtovanju modela je tako iz kosov kot tudi iz sestavov mogoče ustvariti še risbo (angl. Drawing). Tej so samodejno dodani pogledi na model iz različnih smeri, sami pa lahko po potrebi dodamo tudi dimenzije in tolerance. SolidWorks ponuja poleg osnovne aplikacije z najpogosteje uporabljenimi funkcijami še kar nekaj razširitev. Nekatere nam omogočajo na primer simulacije obnašanja modela kot fizičnega objekta v resničnem svetu ali simulacije pri delu s tekočinami. Na voljo so nam tudi razširitve s posebnimi funkcijami za delo z električnimi vezji ali plastiko in orodja za upravljanje podatkov, ki poenostavijo delo v ekipah, kjer več oseb hkrati sodeluje na istem projektu. Za shranjevanje SolidWorks uporablja format dato-

teke, imenovan Microsoft Structured Storage. To pomeni, da je znotraj datoteke s končnico SLDPRT (kos), SLDASM (sestav) ali SLDDRW (risba) v resnici vključenih več različnih datotek, med katerimi so tudi predogled modela in poddatoteke z metapodatki.

S samim modeliranjem se tokrat ne bomo neposredno ukvarjali, zato povejmo le, da je v ta namen na voljo veliko raznovrstnih funkcij in pristopov, o čemer lahko več preberemo v [8] in [9]. Podrobneje pa si oglejmo nekaj drugih pomembnejših funkcionalnosti, ki so nam na voljo in jih bomo potrebovali za realizacijo zastavljenih nalog. Ker bomo imeli veliko opravka z variabilnostjo 3D-modelov, bo za nas ključnega pomena uporaba konfiguracij. Ta nam dovoljuje, da izdelamo več variacij enega kosa ali sestava znotraj istega dokumenta, s čimer lahko ustvarimo družine grafičnih modelov z različnimi dimenzijami, komponentami in parametri. Ko ustvarimo novo konfiguracijo, ji moramo najprej določiti ime in nastavitve, nato pa lahko začnemo z izvajanjem sprememb na modelu in s tem definiramo novo konfiguracijo. Znotraj različnih tipov dokumentov lahko izvedemo različne spremembe. V dokumentu sestava lahko na primer izdelamo novo konfiguracijo tako, da določene komponente skrijemo in na ta način predstavimo poenostavljeno različico modela. Konfiguracije sestava se lahko med sabo razlikujejo tudi po uporabljenih konfiguracijah posameznih kosov, kar bomo kasneje izkoristili tudi v našem algoritmu. Še ena izmed lastnosti, po katerih se lahko razlikujejo konfiguracije, je tudi množica poljubnih lastnosti (angl. Custom properties), ki jih definiramo glede na naše potrebe. Omogočajo nam shranjevanje informacije za kasnejše branje in uporabo v povezavi z določenim modelom oziroma njegovo konfiguracijo, ki to informacijo vsebuje. Dodamo lahko poljubno število vnosov, pri čemer moramo določiti tip vnosa ter ime in vrednost vnesene lastnosti. Med tipi, ki so nam na voljo, so tekst, število, vrednost da/ne ter datum. Vnesene lastnosti nam bodo v našem algoritmu v pomoč pri izdelavi kosovnice, saj lahko s tem nadzorujemo, katere komponente se na njej izpišejo in katere stolpce vsebuje zapis posamezne komponente. Podrobnosti o izdelavi konfiguracij lahko preberemo v [10] in [11].

Ker bomo v naši nalogi vse omenjene funkcionalnosti uporabljali programsko, nadaljujmo z aplikacijskim programskim vmesnikom sistema SolidWorks, ki nam to omogoča.

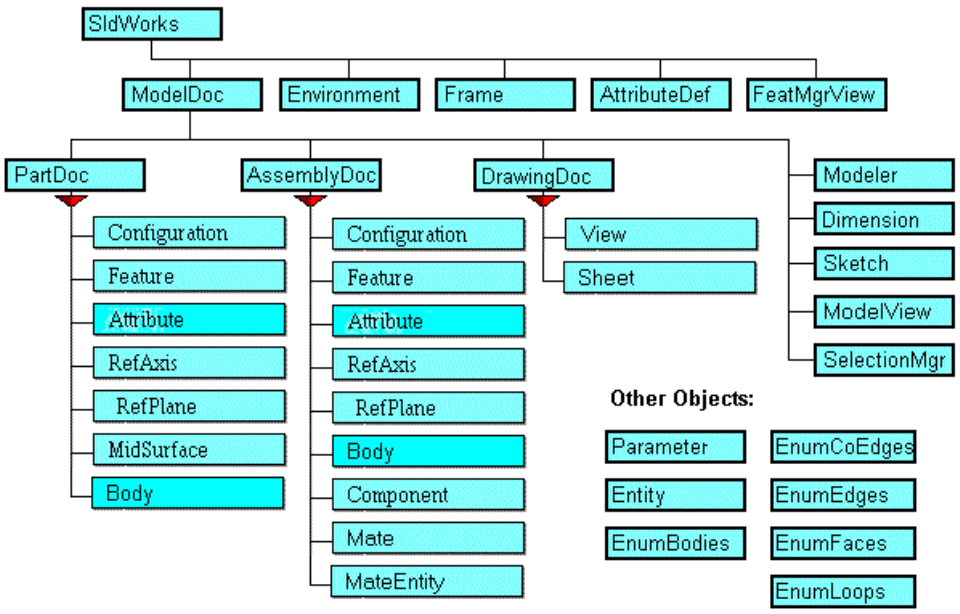
2.4.2 Programski vmesnik

Namestitev orodij za razvoj programske opreme Solidworks API SDK nam omogoči uporabo SolidWorksovega programskega vmesnika (angl. Application programming interface – API), s pomočjo katerega lahko upravljamo delovanje SolidWorksa in njegove podatke. Z uporabo API-ja lahko avtomatiziramo pogosto ponavljajoče se naloge, s čimer standardiziramo njihov rezultat in hkrati odpravimo ročno delo ter s tem možnost napak. Pri tem se poveča produktivnost, prihranimo pa tudi na času in denarju. Poleg tega lahko s pomočjo API-ja razvijemo orodja, specifična za področje, s katerim se ukvarjamo, dodamo lahko nove ukaze in po potrebi tudi spreminjamo nastavitve programa.

Avtomatizacijo nalog in uvedbo novih funkcionalnosti lahko dosežemo na različne načine. Najlažji in najhitrejši pristop je uporaba makroja, ki vsebuje ustrezne API-klice glede na želene akcije v uporabniškem vmesniku SolidWorksa. Makroje lahko napišemo in kasneje urejamo ter poganjamo z uporabo orodja Microsoft VBA (angl. Visual Basic for Applications), ki je del programskega paketa SolidWorks. Makro je shranjen v datoteki s končnico SWP. Enako lahko storimo tudi z orodjem Microsoft VSTA (angl. Visual Studio Tools for Applications), ki pa poleg urejanja VB.NET-kode podpira tudi makroje v programskem jeziku C#. V tem primeru je končni rezultat datoteka s končnico DLL, ki jo lahko v SolidWorksu uporabimo na enak način kot datoteko SWP. Ko imamo makro izdelan, ga lahko dodelimo novemu gumbu v orodni vrstici SolidWorksa in z njim uporabljamo funkcionalnost, ki smo si jo zamislili. Za kompleksnejše naloge je v primerjavi z makrojem verjetno boljša izbira izdelava samostojne aplikacije ali dodatka za SolidWorks, a je za to potrebnega nekoliko več znanja in dela. V ta namen lahko uporabimo katerega koli od programskih jezikov, ki podpirajo COM-vmesnik (angl. Component Objet Model), najpogosteje uporabljeni pa so Visual Basic .NET, Visual C++ in Visual C#.

Preden se lotimo razvoja, moramo poskrbeti še za reference do SolidWorksove knjižnice, ki nam omogoča interakcijo z glavno aplikacijo in njenimi objekti ter izvajanje akcij nad njimi. SldWorks je objekt na najvišjem nivoju in predstavlja aplikacijo SolidWorks. Je vmesnik, prek katerega posredno ali neposredno pridemo do vseh drugih vmesnikov, ki so nam na voljo na nižjih nivojih. To je razvidno tudi iz diagrama objektnega modela na spodnji sliki 2.1.

Če želimo uporabiti objekt, ki ni neposredno dostopen z najvišjega nivoja, moramo najprej poiskati objekt, ki je v hierarhiji nad objektom, do katerega želimo dostopati. Tak primer je recimo objekt Configuration, ki ne more obstajati sam zase in ga lahko dosežemo le posredno. Zato moramo najprej izvesti dostop do objekta ModelDoc, saj Configuration obstaja le znotraj njegovega konteksta. Večina objektov SolidWorks API-ja predstavlja določeno funkcionalnost v uporabniškem vmesniku, nekatere akcije pa so možne le z uporabo API-ja. Za primer vzemimo objekt Feature, ki omogoča dostop do funkcij v načrtovalskem drevesu FeatureManager. Za razliko od njega je objekt Attribute dostopen le preko API-ja, saj v uporabniškem vmesniku ne obstaja korespondenčna funkcionalnost. [12]



Slika 2.1: Objektni model programskega vmesnika SolidWorks.

2.4.3 Dodatek

Dodatek (angl. Add-in) za SolidWorks nam omogoča razširitev obstoječega uporabniškega vmesnika glede na naše potrebe. V menijsko vrstico lahko na primer dodamo nov spustni meni, od koder prožimo različne akcije, ali pa dodamo nov zavihek v orodno vrstico in na njem razporedimo gumbe, ki jih bomo potrebovali

za upravljanje funkcij dodatka. Če želimo dodati funkcionalnost, ki zahteva interakcijo z uporabnikom, lahko oblikujemo tudi razdelek za upravljanje lastnosti (angl. Property manager page). Na njem lahko uporabimo klasične gradnike, kot so gumb, tekstovno polje, seznam itd., in prek njih vplivamo na delovanje programa.

Veliko dela pri razvoju nam lahko prihrani uporaba predloge C# za izdelavo dodatka za SolidWorks, saj ta že poskrbi za kreiranje nekaterih datotek in razredov:

- datoteka AssemblyInfo.cs vsebuje informacije o dodatku, ki so potrebne za organizacijo tega in na koncu tudi za objavo;
- v datoteki EventHandling.cs se nahajajo objekti, ki skrbijo za rokovanje z dogodki (angl. Event handling). Objekt DocumentEventHandler je osnovni razred, objekti PartEventHandler, AssemblyEventHandler in DrawingEventHandler pa razširjajo ta razred in skrbijo vsak za dogodke na določenem tipu datoteke;
- PMPHandler.cs vsebuje istoimenski razred, ki uporablja vmesnik rokovalnika za PropertyManagerPage. To je razdelek na levi strani SolidWorksovega okna, ki nam omogoča urejanje raznih lastnosti dokumenta. Ko sprožimo dogodek v tem razdelku, se nanj odzove ustrezna metoda v rokovalniku;
- razred UserPMPage v datoteki UserPMPage.cs je ovojni razred (angl. Wrapper class), ki omogoča izdelavo razdelka PropertyManagerPage po uporabnikovih željah in potrebah. Omogoča dodajanje, pozicioniranje in urejanje videza kontrol, kot so na primer gumbi, tekstovna polja in seznami;
- datoteka SwAddin.cs vsebuje glavni razred dodatka za SolidWorks, ki se deli na več segmentov:
  - Local Variables – definicija razrednih spremenljivk;
  - SolidWorks Registration – metode, ki skrbijo za registracijo dodatka, da ga SolidWorks med svojim izvajanjem prepozna;
  - ISwAddin Implementation – vstopna in izstopna točka dodatka;
  - UI Methods – definicija delovanja dodatka znotraj aplikacije in osnovni izgled oziroma postavitev;

- Event Methods – metode, ki inicializirajo rokovanje z dogodki;
- Event Handlers – metode za rokovanje z dogodki;
- UI Callbacks – metode, ki predstavljajo akcije ob uporabi uporabniškega vmesnika dodatka.

Z uporabo predloge lahko tako že vzpostavimo osnovno strukturo dodatka za SolidWorks, ki ga je v tem stanju tudi že mogoče registrirati, da ga SolidWorks prepozna in vključi v delovanje osnovne aplikacije. [13] Seveda tukaj še nimamo prave funkcionalnosti, večino te pa bomo dodali znotraj segmenta UI Callbacks v datoteki SwAddin.cs. Izgled oziroma postavitev kontrol, ki prožijo različne akcije, pa bomo definirali v segmentu UI Methods.

2.5 CiciVO

Program ciciVO služi kot podporno orodje pri prodaji vgradnih omar. Je del programskega paketa ciciWORKS, razvitega v podjetju IB-CADDY, d. o. o., ki vključuje tudi 3D-pregledovalnik, imenovan ciciVIEW.

CiciVO omogoča ustvarjanje nove konfiguracije vgradne omare kot tudi urejanje že obstoječe. Z njim lahko nastavljamo številne parametre, ki določajo končno podobo vgradne omare, in ga tako do potankosti prilagodimo svojim željam. Poglejmo si nekaj glavnih funkcionalnosti. V osnovi ciciVO loči tri tipe vgradnih omar: prvi je tip I, ki predstavlja klasično vgradno omaro za postavitev ob steno, tip L označuje kotno vgradno omaro, tip U pa je prav tako kotna omara, le da pokriva dva sosedna kota v prostoru. Ko se odločimo za tip omare, običajno načrtovanje nadaljujemo z nastavitvijo zunanjih dimenzij, ki nam določijo, katere nadaljnje prilagoditve nam bodo na voljo. V prvi vrsti je od dimenzij odvisna izbira števila vratnih kril. Za vsako krilo posebej si lahko nato zamislimo še delitev polnil in njihovo barvo ali teksturo. Z izbiro števila vratnih kril določimo tudi število segmentov, na katere je razdeljena notranjost. Te lahko zapolnimo z vnaprej pripravljenimi razporeditvami notranjih elementov in jih nato lahko po lastnih potrebah tudi prilagodimo. Izbiramo lahko med kombinacijami polic, predalov in različnih obešalnikov. Medtem ko spreminjamo parametre vgradne omare, lahko trenutni videz te spremljamo v vključenem 3D-pregledovalniku, kar nam je v veliko

pomoč pri načrtovanju. Končni rezultat je projektna datoteka s končnico CWP, ki bo tudi vhodna datoteka našega algoritma. V njej je zapisana celotna drevesna struktura, ki je zamišljena tako, da korenski objekt Assembly vsebuje morebitne podsestave v obliki objektov tipa Component, ti pa nato nosijo referenco na nov objekt tipa Assembly. Če Assembly ne vsebuje nobene komponente, gre za list v drevesni strukturi. Ker v SolidWorksu prav tako obstajata razreda Assembly in Component, povejmo, da govorimo o razredih, definiranih v knjižnici programskega paketa ciciWORKS, ki zna rokovati z objekti, opisanimi v datoteki CWP. Imenom teh bomo zato v našem algoritmu dodali predpono »cw«, objektom SolidWorks pa »sw«.

Za predstavitev geometrije vgradne omare ciciVO uporablja 3D grafično knjižnico OpenSceneGraph, ki jo bomo natančneje opisali v naslednjem poglavju.

2.6 OpenSceneGraph

OpenSceneGraph (OSG) je odprtokodni 3D-grafični aplikacijski programerski vmesnik (API), ki se pogosto uporablja za razvoj visoko zmogljivih grafičnih aplikacij na področjih, kot so simulatorji letenja, računalniške igre, vizualne simulacije, navidezna resničnost, znanstvene in medicinske vizualizacije ter modeliranje. Napisan je v standardnem jeziku C++ z uporabo knjižnice Standard Template Library (STL). Za predstavitev 3D-sveta uporablja pristop grafa scene in nudi objektno orientirano ogrodje nad knjižnico OpenGL, s čimer poskrbi za implementacijo in optimizacijo nizkonivojskih grafičnih klicev ter s tem razbremeni razvijalca programske opreme. Poleg tega je na voljo tudi veliko orodij, ki omogočajo hiter razvoj grafičnih aplikacij. Namen OSG-ja je narediti tehnologijo grafa scene in njene prednosti dostopne vsem uporabnikom, tako komercialnim kot tudi nekomercialnim.

Graf scene predstavlja podatkovno strukturo, ki opisuje logično in pogosto tudi prostorsko predstavitev grafične scene. Je zbirka vozlišč v grafu ali drevesni strukturi, ki imajo običajno veliko naslednikov in le eno starševsko vozlišče. To omogoča izvajanje operacij na celotni skupini, kar avtomatsko vpliva na njene elemente. V mnogih grafičnih aplikacijah ima vsak nivo v drevesu svojo transformacijsko matriko, ki predstavlja položaj glede na predhodno vozlišče. Položaj glede na druge nivoje kot tudi globalni položaj tako dobimo z množenjem matrik vseh nivojev. Po-

gosta operacija, ki izkorišča ta pristop, je možnost ustvarjanja skupine več objektov, ki so v relaciji med seboj, in izvajanje operacij nad njo kot nad enim objektom. Tako lahko na primer celotno skupino hkrati premikamo in izvajamo druge transformacije. Glavne prednosti OSG-ja so učinkovitost, skalabilnost in prenosljivost, skozi celotno programsko kodo pa se ponavljajo enaki strukturni vzorci, kar jo naredi tudi preprostejšo za uporabo in vzdrževanje ter olajša razumevanje delovanja. Na voljo je tudi veliko dobrih primerov uporabe. Zaradi modularnosti in razširljivosti je končnim uporabnikom omogočeno, da uporabijo le komponente, ki jih dejansko potrebujejo, in uvedejo spremembe, kjer je to potrebno.

OSG nudi podporo za OpenGL od verzije 1.0 naprej ter OpenGL ES 1.1 in 2.0, kar omogoča uporabo tako s starejšo strojno opremo in operacijskimi sistemi kot tudi z najnovejšimi mobilnimi napravami in vsemi najzmogljivejšimi namiznimi grafičnimi sistemi. Poganjamo ga lahko na različnih operacijskih sistemih, vključno z Microsoft Windows, Mac OS X, Linux, IRIX, Solaris in FreeBSD, od verzije 3.0.0 naprej pa OSG nudi podporo tudi za razvoj aplikacij na mobilnih platformah Android in iOS. [14]

Poglavje 3

Programska rešitev

3.1 Vhodni podatki

Da bo algoritem pravilno deloval in opravljal svojo funkcijo gradnje 3D-modela, moramo najprej poskrbeti za vse vhodne podatke. Najpomembnejši del je verjetno datoteka CWP, v katero program ciciVO shrani svoj projekt in vsebuje strukturo ter vse parametre, potrebne za opis končne podobe vgradne omare. Datoteka CWP je arhivska datoteka, ki združuje geometrijski in prodajni del podatkov. Oba uporabljata funkcionalnost datotek OSG, ki smo jih omenili v prejšnjem poglavju. Prvi del je shranjen v obliki OSG ASCII-formata v datoteki s končnico OSGT. Z referenco na binarno datoteko OSGB je tu definiran osnovni oziroma bazni tip vgradne omare (I, L, U). To je korenska datoteka naše drevesne strukture. Celotna geometrija je nato zgrajena z referencami na datoteke nižjih nivojev, vendar pa tega ne bomo podrobneje obravnavali, saj ni ključnega pomena za to diplomsko nalogo. Poleg definicije baznega tipa je v datoteki OSGT prisoten še seznam vrednosti vseh parametrov, ki smo jih nastavili v uporabniškem vmesniku programa cicVO in tako določili videz vgradne omare. Približna struktura te datoteke je prikazana v spodnjem izskeu kode, kjer lahko vidimo izbrani bazni tip I in nato seznam parametrov.

```
#Ascii Object
#Version 111
#Generator OpenSceneGraph 3.3.1

Project {
  UniqueID 1
  DocumentUri "C:/.../I_CLOSET.osgb"
  Object {
    UniqueID 2
    ReadOnly FALSE
    ...
    ExternalReference "./data.osgx"
  }

  Parameters 109 {
    "PARAM001@Doorleaf 001@Door@I_Closet" {
      UniqueID 3
      Key "PARAM001"
      Value {
        Type Double
        Variant 0
      }
    }
    ...
    "PARAM097@Segment 002@Korpus@I_Closet" {
      UniqueID 97
      Key "PARAM097"
      Value {
        Type String
        Variant "value"
      }
    }
    ...
  }
}
```

Izsek kode 3.1: Opis geometrije v datoteki OSGT

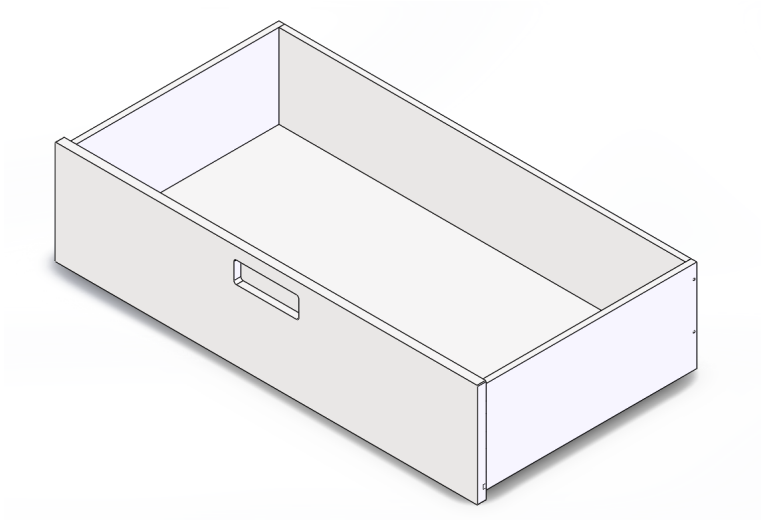
Prodajni del se nahaja v datoteki OSGX in nosi informacijo o šifrah posameznih komponent, uporablja pa se tudi za izračun cen. Majhen izsek iz te datoteke lahko vidimo v nadaljevanju.

```
<?xml version="1.0" encoding="UTF-8" ?>
<Object>
  <Version attribute="111" />
  <Generator attribute="OpenSceneGraph 3.3.1" />
  <UniqueID attribute="1" />
  <Exchange>
        ...
```

```
        <Assembly>
          <Index attribute="0" />
          <Name attribute="DNO" />
          <Type attribute="BOTTOM" />
          <Parameters attribute="3">
            ...
          </Parameters>
          <Articles attribute="1">
            <Article>
              <Ident attribute="CB-1" />
              <Name attribute="CARCASE_BOTTOM" />
              <Qty attribute="1" />
              <Price attribute="12.79" />
              <PriceWithFactor attribute="12.79" />
              <VAT attribute="0" />
              <Unit />
              <Parameters attribute="2">
                <Parameter>
                  <Name attribute="WidthR" />
                  <Key attribute="033" />
                  <Value attribute="815" />
                </Parameter>
                <Parameter>
                  <Name attribute="DepthR" />
                  <Key attribute="035" />
                  <Value attribute="480" />
                </Parameter>
              </Parameters>
            </Article>
          </Articles>
        </Assembly>
    ...
  </Exchange>
</Object>
```

Izsek kode 3.2: Prodajni podatki v datoteki OSGX

Drugi del vhoda je knjižnica datotek SolidWorks, ki predstavlja nabor vseh možnih komponent vgradne omare. Od tega je velika večina kosov, modeli nekaterih komponent pa so že izdelani sestavi in jih lahko uporabimo pri gradnji. Ker so dimenzije večine komponent odvisne od nastavitev, ki jih uporabnik definira v programu ciciVO, morajo biti modeli v knjižnici izrisani tako, da njihove dimenzije določajo parametri z enakimi imeni, kot so definirani v vhodni datoteki CWP. Vsi modeli na začetku vsebujejo eno privzeto konfiguracijo, med gradnjo pa dodajamo nove z ustreznimi vrednostmi parametrov. Zato moramo poskrbeti, da imajo datoteke nastavljeno dovoljenje za pisanje.



Slika 3.1: Primer sestava v knjižnici datotek SolidWorks.

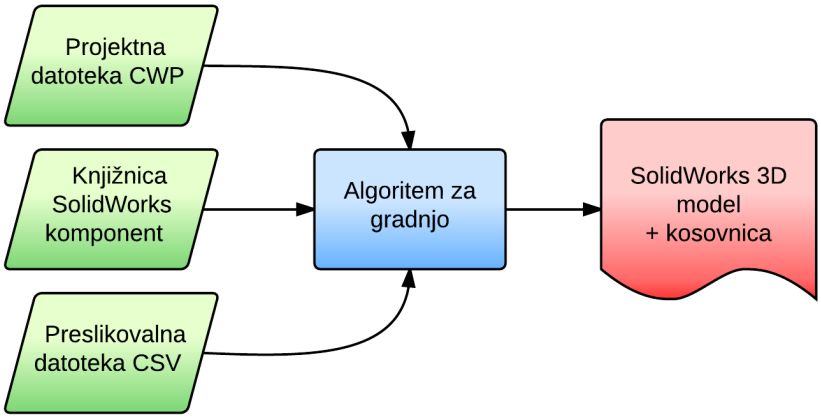
Za vsak tip komponente v projektni datoteki CWP se v knjižnici nahaja pripadajoči 3D-model. Te preslikave definira datoteka CSV, ki je naša zadnja vhodna datoteka. Vsak vnos vsebuje tudi tip datoteke 3D-modela, ki nam pove, ali pri gradnji vstavljamo kos ali sestav. Za nekatere komponente si včasih želimo, da bi jih izključili iz kosovnice, zato je v preslikovalni datoteki prisotna tudi ta informacija. Za druge pa lahko, če je to potrebno, spremenimo ime, ki se izpiše na kosovnici. Datoteka CSV nam omogoča, da v primeru nadgradnje dodamo poljubno število novih parametrov in tako preprosto razširimo funkcionalnost. V tabeli 3.1 lahko vidimo nekaj primerov vnosa v to datoteko.

S tem imamo pripravljene vse potrebne vhodne podatke.

compType	id	docType	excludeFromBOM	nameOverride
SHELF	SH-10	PRT	N	
DRAWER	DR-9	ASM	N	
DOORLEAF	DL-12	PRT	N	Vratno krilo

Tabela 3.1: Primeri vnosa v preslikovalni datoteki CSV

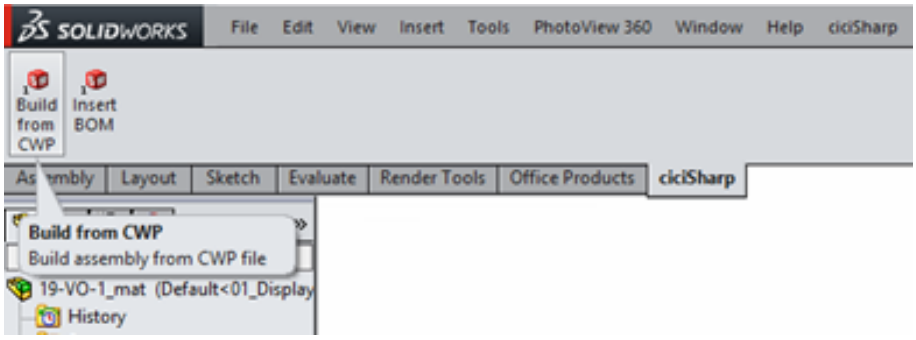
Ideja naše programske rešitve je zgraditi SolidWorksov sestav, ki bo predstavljal 3D-model vgradne omare, opisane v projekti datoteki CWP. Pri tem s pomočjo preslikovalne datoteke CSV izberemo ustrezne komponente iz knjižnice in jih vstavimo v sestav. Na koncu v dokumentu sestava oblikujemo še kosovnico.



Slika 3.2: Ideja delovanja programske rešitve.

3.2 Uporabniški vmesnik

Funkcionalnost dodatka za SolidWorks smo se odločili razdeliti na dva dela, in sicer na gradnjo 3D-modela ter izpis kosovnice. V ta namen smo v orodni vrstici ustvarili nov zavihek z dvema gumboma, ki predstavljata vsak svoj del funkcionalnosti.



Slika 3.3: Izgled uporabniškega vmesnika.

Sledi nekaj pomembnejših metod, ki smo jih uporabili za izdelavo uporabniškega vmesnika na zgornji sliki.

S pomočjo funkcije v izseku 3.3 ustvarimo skupino ukazov, ki imajo enake nastavitve, kot na primer prisotnost v orodni vrstici in spustnem meniju. Kot parametre ji moramo podati enolični identifikator, ime in opis, ki se izpiše ob lebdenju miške nad skupino, ter namig v statusni vrstici. Podamo tudi pozicijo in indikator, ali naj se razveljavijo nastavitve, ki so veljale do zdaj. Zadnji je izhodni parameter, ki dobi vrednost kode napake, če je med izvajanjem do te prišlo.

```
CommandGroup CreateCommandGroup2(
   System.int UserID,
   System.string Title,
   System.string ToolTip,
   System.string Hint,
   System.int Position,
   System.bool IgnorePreviousVersion,
   out System.int Errors
)
```

Izsek kode 3.3: Funkcija CreateCommandGroup2

S klicem programske kode v izseku 3.4 dodamo nov ukaz v prej ustvarjeno skupino ukazov. Tukaj določimo ime, pozicijo, namig in opis ukaza ter indeks slike, ki se uporabi kot ikona. Navedemo še ime funkcije, ki se izvede kot povratni klic ob pritisku v uporabniškem vmesniku. Po potrebi lahko tej funkciji podamo tudi parameter tipa string. Ukaz lahko v določenih primerih v uporabniškem vmesniku tudi onemogočimo, kar storimo z navedbo imena funkcije, ki vrača vrednost »bool«, s katero pove, ali je ukaz omogočen ali ne. Na koncu sledi enolični identifikator ukaza in kot zadnji parameter še nastavitve za prikaz ukaza v meniju.

```
System.int AddCommandItem2(
   System.string Name,
   System.int Position,
   System.string HintString,
   System.string ToolTip,
   System.int ImageListIndex,
   System.string CallbackFunction,
   System.string EnableMethod,
   System.int UserID,
   System.int MenuTBOption
)
```

Izsek kode 3.4: Funkcija AddCommandItem2

Spodnja funkcija doda v orodno vrstico nov zavihek, ki bo vseboval ukaze za naš dodatek. Podamo ji tip dokumenta, za katerega želimo dodati zavihek (kos, sestav ali risba), in ime samega zavihka.

```
CommandTab AddCommandTab(
   System.int DocumentType,
   System.string TabName
)
```

Izsek kode 3.5: Funkcija AddCommandTab

Funkcijo AddCommandTabBox pokličemo nad objektom ustvarjenega zavihka in mu dodamo nov razdelek oziroma skupino ukazov. Funkcija ne sprejme nobenega parametra. Skupina, ki je ustvarjena, je vedno pozicionirana kot zadnja v zavihku. Če želimo spremeniti vrstni red skupin, moramo sami prilagoditi vrstni red klicev te funkcije.

```
CommandTabBox AddCommandTabBox()
```

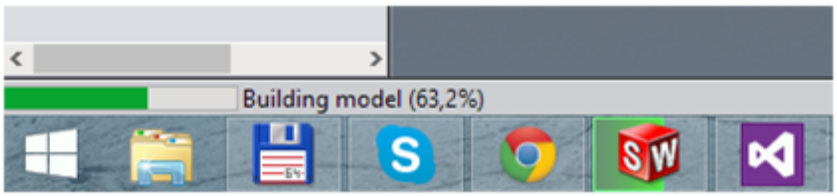
Izsek kode 3.6: Funkcija AddCommandTabBox

Na koncu moramo ukaze dodati v določen razdelek v zavihku orodne vrstice. Ko nad objektom razdelka pokličemo funkcijo v izseku 3.7, ji podamo parameter s seznamom identifikatorjev ukazov, ki jih želimo dodati. Za vsak ukaz se v razdelku ustvari nov gumb z imenom, ki smo ga navedli pri dodajanju ukaza s funkcijo AddCommandItem2. Drugi parameter je prav tako seznam in za vsak identifikator v prvem seznamu določa pozicijo teksta na gumbu. Ta je lahko pod ikono ali ob njej, lahko pa je tudi skrit.

```
System.bool AddCommands(
   System.object CommandIDs,
   System.object TextDisplayStyles
)
```

Izsek kode 3.7: Funkcija AddCommands

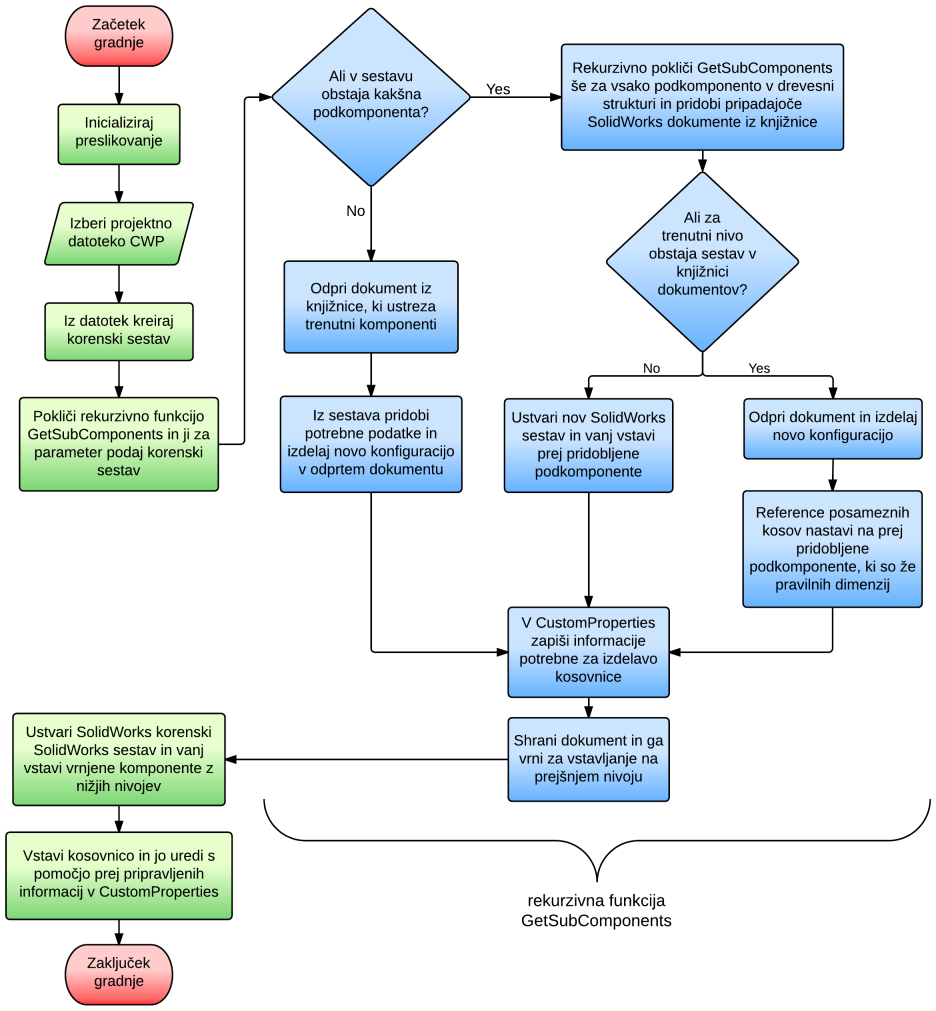
Ustvarili smo torej gumba »Build from CWP« ter »Insert BOM«, ki sprožita vsak svojo funkcijo, in s tem je naš uporabniški vmesnik pripravljen za uporabo. Je sicer zelo preprost, saj gre za avtomatizacijo procesa gradnje 3D-modela in zato algoritem ne potrebuje skoraj nobene interakcije z uporabnikom. Potrebna sta le začetni pritisk na gumb in izbira vhodne datoteke CWP, nato pa poteka samodejna gradnja 3D-modela. Med izvajanjem algoritma lahko v levem spodnjem kotu uporabniškega vmesnika spremljamo tudi indikator napredka.



Slika 3.4: Indikator napredka.

Zdaj je treba napisati le še funkcije, ki smo jih prej navedli kot povratne klice ob pritisk na posamezen gumb.

3.3 Algoritem za gradnjo



Slika 3.5: Diagram poteka algoritma za gradnjo.

Vstopna točka našega algoritma za gradnjo je funkcija BuildFromCWP. Njena prva naloga je, da sproži inicializacijo preslikovanja komponent. Pokliče se funkcija InitializeMapping, ki ustvari objekt tipa Dictionary, ki bo služil kot preslikovalni slovar. Napolnimo ga z vnosi iz preslikovalne datoteke CSV. Ključ za iskanje po slovarju je tip komponente, njegova pripadajoča vrednost pa so informacije za preslikavo (ime

datoteke v knjižnici in njen tip). Nato odpremo dialog za izbiro datoteke, kjer izberemo projektno datoteko CWP, iz katere želimo zgraditi 3D-model v SolidWorksu. Vsebino izbrane datoteke preberemo v objekt cwRootAssembly tipa Assembly, ki predstavlja vrh drevesne strukture naše geometrije oziroma korenski sestav. Tega nato kot parameter cwAssembly podamo rekurzivni funkciji GetSubComponents, ki vsebuje glavno funkcionalnost našega algoritma.

```
Assembly cwRootAssembly =
    Assembly.readFromFile(openFileDialog.FileName);

private Dictionary<string, object> GetSubComponents(Assembly
    cwAssembly)
```

Izsek kode 3.8: Klic rekurzivne funkcije GetSubComponents

Funkcija GetSubComponents se v zanki sprehodi skozi vse podkomponente in za vsako, spet kot parameter cwAssembly, poda novemu klicu funkcije GetSubComponents. Tudi na naslednjem in morebitnih še nižjih nivojih ponovimo postopek, s čimer pridemo počasi do konca ene izmed vej drevesa naše geometrije.

```
foreach (Component cwSubComponent in cwAssembly.Components)
{
    Assembly cwSubAssembly = cwSubComponent.getAssembly();
    swSubComponents = GetSubComponents(cwSubAssembly);

    if (swSubComponents != null)
    {
        names.AddRange((subComponents["names"]
            as List<string>));
        transforms.AddRange((subComponents["transforms"]
            as List<double>));
        coordSystems.AddRange((subComponents["coordSystems"]
            as List<string>));
        configNames.AddRange((subComponents["configNames"]
            as List<string>));
    }
}
```

Izsek kode 3.9: Zanka za rekurzivno obdelavo podkomponent

Znotraj zanke torej v vsakem obhodu izvedemo rekurzivni klic. Ko se ta izvede, to pomeni, da smo obdelali eno poddrevo tega nivoja in hkrati že zgradili pripadajočo strukturo v SolidWorksu. Enako moramo storiti še s preostalimi poddrevesi.

Zato SolidWorksove kose in sestave oziroma njihove poti, ki jih vrne rekurzivni klic GetSubComponents, shranimo na seznam in začnemo nov obhod zanke ter s tem obdelavo naslednjega poddrevesa.

3.3.1 Obdelava listov drevesne strukture

Z gradnjo 3D-modela v SolidWorksu začnemo na najnižjem nivoju. Kot smo povedali, listi drevesne strukture ne vsebujejo novih komponent, zato se prej opisana zanka v tem primeru ne izvede. Lotimo se torej obdelave najnižje komponente v strukturi, in sicer za pripadajoči objekt cwAssembly preverimo, kakšnega tipa je, ter poiščemo ustrezen vnos v preslikovalnem slovarju.

```
string asmType = cwAssembly.getType();
AsmTypeMapping asmTypeMap = _asmTypeMap[asmType];
string name = asmTypeMap.Name;
```

Izsek kode 3.10: Uporaba preslikovalnega slovarja

Z izvedbo izseka kode 3.10 dobimo ime komponente v knjižnici datotek SolidWorks, ki predstavlja geometrijo obravnavanega nivoja. Ker je ta privzetih dimenzij, skoraj zagotovo ni primerna za uporabo v točno tej vgradni omari. Zato je treba ustvariti novo konfiguracijo z ustreznimi dimenzijami. To storimo tako, da odpremo dokument in pokličemo funkcijo SetConfiguration, ki poskrbi za nastavitev pravilne konfiguracije. Informacije, ki jih ta potrebuje, dobimo s sprehodom čez parametre objekta cwAssembly.

```
ModelDoc2 swModelDoc = iSwApp.OpenDoc6(
    componentPath,
    (int)swDocumentTypes_e.swDocPART,
    (int)swOpenDocOptions_e.swOpenDocOptions_Silent,
    "", ref errors, ref warnings
);

Configuration swConfig = SetConfiguration(swModelDoc,
    cwAssembly);
```

Izsek kode 3.11: Klic funkcije SetConfiguration

Če je bila ustrezna konfiguracija že kdaj uporabljena, jo aktiviramo, sicer pa se ustvari nova. Izbrani konfiguraciji nastavimo še nekaj poljubnih lastnosti (Custom Properties), ki jih bomo kasneje uporabili za izpis na kosovnici.

```
Configuration swConfig =
    swModelDoc.GetConfigurationByName(configName);

if (swConfig == null)
    swConfig = swModelDoc.AddConfiguration3(configName, "", "",
        0);

swConfig.CustomPropertyManager.Add3(
    "excludeFromBOM",
    (int)swCustomInfoType_e.swCustomInfoText,
    asmTypeMap.ExcludeFromBOM.ToString(),
    (int)swCustomPropertyAddOption_e.swCustomPropertyDeleteAndAdd
);
```

Izsek kode 3.12: Dodajanje nove konfiguracije

Na koncu dokument shranimo ter njegovo pot, ime izbrane konfiguracije in transformacijo (relativno na en nivo višje) vrnemo kot rezultat funkcije GetSubComponents ter nadaljujemo z izvajanjem na višjih nivojih.

```
swModelDoc.Extension.SaveAs(
    newComponentPath,
    (int)swSaveAsVersion_e.swSaveAsCurrentVersion,
    (int)swSaveAsOptions_e.swSaveAsOptions_Silent,
    null, ref errors, ref warnings
);

Dictionary<string, object> returnList =
    new Dictionary<string, object>();

returnList.Add("names", names);
returnList.Add("transforms", transforms);
returnList.Add("coordSystems", coordSystems);
returnList.Add("configNames", configNames);

return returnList;
```

Izsek kode 3.13: Vračanje rezultata

3.3.2 Obdelava višjih nivojev

Predpostavimo, da ima sestav na predzadnjem nivoju pet podkomponent. To pomeni, da ima pet poddreves oziroma v tem primeru listov drevesa – vsakega posebej obdelamo v petih obhodih že prej opisane zanke. V vsakem obhodu za posamezno komponento izvedemo ugnezden rekurzivni klic funkcije GetSubComponents, ki nam vrne pot do kosa SolidWorks, tako kot smo to opisali v prejšnjem odstavku. Vrnjeno pot dodamo na seznam podkomponent tega nivoja in nadaljujemo z naslednjim obhodom zanke, kjer postopek ponovimo. Ko obdelamo vsa poddrevesa, imamo na seznamu podkomponent poti do petih kosov SolidWorks. Poleg tega imamo še seznam z imeni pripadajočih ustreznih konfiguracij in enega s pripadajočimi transformacijami komponent.

Na tem mestu se algoritem razdeli v dve veji, glede na to, ali je trenutno obravnavana komponenta v knjižnici datotek SolidWorks prisotna kot sestav ali ne. To preverimo v preslikovalnem slovarju.

Če ugotovimo, da je iskani sestav v knjižnici prisoten, moramo vrnjene podkomponente, shranjene na seznamu, obravnavati kot dele tega sestava. Zato ga odpremo, ustvarimo novo konfiguracijo in v tej popravimo reference kosov na tiste s seznama podkomponent.

```
iSwApp.ReplaceReferencedDocument(
    swAssemblyPath,
    externalReference,
    newExternalReference
);
```

Izsek kode 3.14: Zamenjava referenčnih dokumentov

Ker smo jih obravnavali že na enem nivoju nižje, je pri vseh kosih že prisotna ustrezna konfiguracija, ki jo je treba le aktivirati z nastavitvijo lastnosti ReferencedConfiguration.

```
swComponent.ReferencedConfiguration = configNames[index];
```

Izsek kode 3.15: Uporaba ustrezne konfiguracije

Ko smo to uredili, enako kot na nivoju listov, v ustvarjeno konfiguracijo odprtega sestava, zapišemo še nekaj lastnosti za kasnejši prikaz na kosovnici. Dokument

shranimo ter tudi tukaj njegovo pot, ime konfiguracije in transformacijo vrnemo na en nivo višje.

Postopek je nekoliko preprostejši, če sestava v knjižnici ni. V tem primeru moramo ustvariti nov SolidWorksov sestav in vanj vstaviti že pripravljene podkomponente. To storimo tako, da nad ustvarjenim sestavom pokličemo funkcijo AddComponents3 ter ji kot parametre podamo seznam poti do podkomponent, seznam transformacij in seznam koordinatnih sistemov. Slednji je sicer seznam praznih stringov, saj ves čas uporabljamo privzeti koordinatni sistem.

```
AssemblyDoc swSubAssembly = iSwApp.NewDocument(
    swAssemblyTemplate,
    (int)swDwgPaperSizes_e.swDwgPaperA4size,
    0.0, 0.0
);

object[] addedComponents = swSubAssembly.AddComponents3(
    names.ToArray(),
    transforms.ToArray(),
    coordSystems.ToArray()
);
```

Izsek kode 3.16: Vstavljanje podkomponent

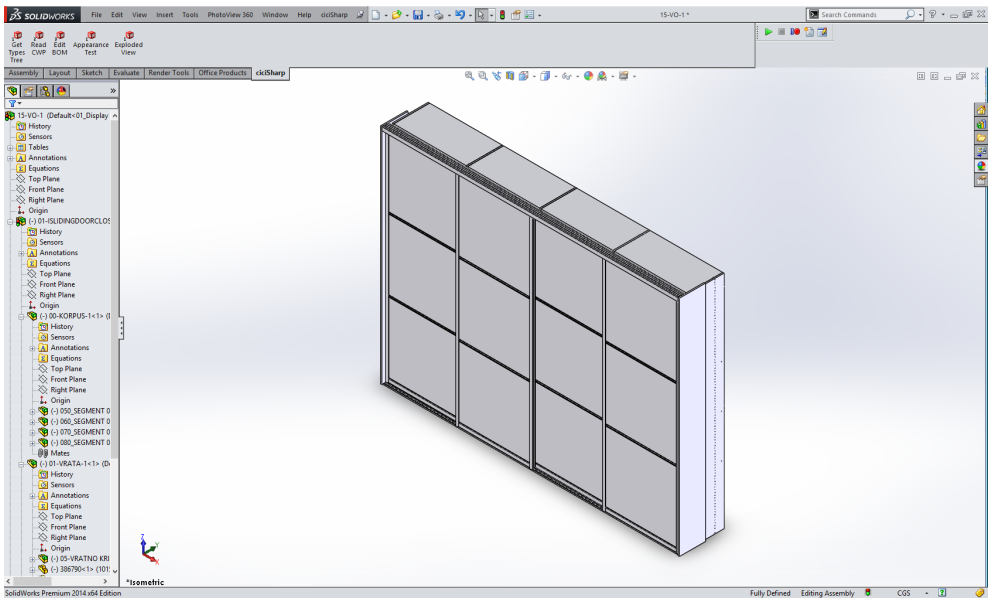
Rezultat funkcije je seznam komponent, ki so bile vstavljene v sestav. V zanki se sprehodimo čez njih in vsaki nastavimo vrednost ReferencedConfiguration na tisto iz seznama ustreznih konfiguracij. Vsi omenjeni seznami so seveda sortirani v enakem vrstnem redu v smislu komponent, na katere se nanaša informacija na določenem indeksu seznama.

```
Component2 swComponent = null;
string configName = null;

for (int i = 0; i < addedComponents.Length; i++)
{
    swComponent = addedComponents[i] as Component2;
    configName = configNames[i];

    if (!String.IsNullOrEmpty(configName))
        swComponent.ReferencedConfiguration = configName;
}
```

Izsek kode 3.17: Nastavljanje konfiguracije dodanim komponentam

Tudi v tem primeru na koncu nastavimo še lastnosti za izpis na kosovnici, shranimo dokument ter na višji nivo vrnemo pot do sestava, ime konfiguracije in transformacijo.

Ko enako obdelamo najvišji nivo, vrnemo rezultat v funkcijo BuildFromCWP, od koder smo izvedli začetni rekurzivni klic GetSubComponents. Tam ustvarimo korenski sestav SolidWorks in vanj vstavimo vrnjene podkomponente, dokument shranimo in zaključimo gradnjo. S tem dobimo SolidWorksovo predstavitev naše vgradne omare, ki smo jo izdelali v programu ciciVO. Nad 3D-modelom lahko zdaj izvajamo vse CAD-funkcije, ki jih ponuja SolidWorks, vključno z izdelavo kosovnice.



Slika 3.6: Vgradna omara.

3.3.3 Izdelava kosovnive

Čeprav izdelava kosovnice v SolidWorksu ni zapleten postopek, smo se odločili tudi ta del avtomatizirati in s tem standardizirati rezultat ter prihraniti še nekaj dodatnega časa. Funkcijo InsertBOM, ki poskrbi za izdelavo, bi lahko poklicali že ob koncu izvajanja BuildFromCWP, vendar smo se za potrebe diplomske na-

loge odločili funkcionalnost ločiti. Kot smo nekajkrat omenili, smo v konfiguracije dokumentov SolidWorks shranili nekatere informacije, ki jih bomo uporabili za izdelavo kosovnice. To so višina, širina in globina, ime komponente ter informacija, ali se komponenta s kosovnice izvzame. Katere stolpce bo tabela kosovnice vsebovala, določimo s predhodno izdelavo predloge, ki jo nato uporabimo pri izdelavi. V našem primeru so prisotni naslednji stolpci: zaporedna številka komponente, šifra, višina, širina, globina in količina. Kosovnico v dokument vstavimo z ukazom InsertBomTable3, ki med drugim kot parameter sprejme ime predloge, način oštevilčevanja in koordinate kosovnice v dokumentu.

```
BomTableAnnotation btAnnotation =
    swModel.Extension.InsertBomTable3(
    templateName,
    0, -250,
    (int)swBomType_e.swBomType_Indented,
    configName,
    false,
    (int)swNumberingType_e.swNumberingType_Detailed,
    true
);
```

Izsek kode 3.18: Vstavljanje kosovnice

Ta zdaj vsebuje vse komponente sestava. Ker nekaterih ne želimo prikazati, se od spodaj navzgor sprehodimo skozi njih in za vsako preverimo, ali jo moramo s kosovnice odstraniti. To nam pove med gradnjo nastavljena poljubna lastnost »excludeFromBOM«.

```
swConfig.CustomPropertyManager.Get5(
    "excludeFromBOM",
    true,
    out excludeFromBOM,
    out excludeFromBOM_resolved,
    out wasResolved
);
```

Izsek kode 3.19: Pridobivanje lastnosti konfiguracije

Prikaz na kosovnici preprečimo tako, da lastnost komponente ExcludeFromBOM v posamezni vrstici nastavimo na true oziroma pokličemo ukaz Dissolve, če gre za komponento, ki je sestav.

```
if (excludeFromBOM)
{
    if (swModelDoc.GetType() ==
        (int)swDocumentTypes_e.swDocASSEMBLY)
        bomTableAnnotation.Dissolve(i); // row i
    else
        rowComponent.ExcludeFromBOM = true;
}
```

Izsek kode 3.20: Izvzemanje komponent s kosovnice

Ostane nam le še shranjevanje dokumenta, s čimer zaključimo izvajanje algoritma.

Poglavje 4

Sklepne ugotovitve

V okviru diplomske naloge smo spoznali prednosti, ki jih pri načrtovanju proizvodnega izdelka nudi uporaba programske opreme CAD. S pomočjo te lahko izdelamo izjemno natančen 3D-model in s tem vizualizacijo končnega izdelka pred dejansko proizvodnjo. Poleg tega nam nudi tudi različna orodja za izdelavo analiz in pripravo dokumentacije. Z vsem tem bistveno zmanjšamo stroške in čas, ki je potreben za proizvodnjo. Eden najpogosteje uporabljenih sistemov CAD je tudi SolidWorks, ki smo si ga podrobneje ogledali in zanj razvili dodatek za avtomatizacijo gradnje 3D-modela. Na ta način smo premostili problem, ki ga predstavlja načrtovanje izdelka z neskončno množico variacij. Kljub pomoči računalniško podprtega načrtovanja bi nam to vzelo ogromno časa, saj bi morali za vsako variacijo izdelka tega na novo izrisati. Algoritem smo sicer razvili za potrebe načrtovanja vgradnih omar, vendar ga je ob ustrezni zamenjavi vhodnih podatkov mogoče uporabiti za avtomatizacijo gradnje katerega koli izdelka. Program ciciVO, ki služi za podporo prodaji vgradnih omar, omogoča veliko prilagoditev po naročnikovih željah. Ker na končno podobo vgradne omare vplivajo številni dejavniki, kot so dimenzije in razporeditev elementov notranjosti glede na želje naročnika, si običajno nista povsem enaki niti dve vgradni omari. Z uporabo našega dodatka za SolidWorks lahko za poljubno omaro v nekaj minutah izdelamo 3D-model, ki je tako že pripravljen za proizvodnjo. V primerjavi z vsakokratnim ročnim načrtovanjem je prihranek na času očiten.

Kljub temu da smo dosegli zastavljeni cilj, pa je še vedno prostor za nadaljnji razvoj. V mislih imamo že nekaj nadgradenj, ki bi še dodatno prispevale k

uporabnosti našega dodatka za SolidWorks. Ena izmed njih je vključitev tekstur v gradnjo 3D-modela vgradne omare. Algoritem bi bilo treba razširiti tako, da ob nastavljanju dimenzij posamezne komponente določimo tudi ustrezno teksturo. Vso potrebno informacijo že imamo v projektni datoteki CWP, treba jo je le pravilno interpretirati in uporabiti.

Druga možna razširitev funkcionalnosti dodatka je izdelava eksplozijskega pogleda celotne vgradne omare in njenih posameznih delov. Končne podobe vgradna omara seveda ne dobi v proizvodnji, saj je potrebna še montaža. S pomočjo eksplozijskega pogleda si lažje predstavljamo, kam spada kakšna komponenta. To bi olajšalo delo monterja, saj smo že večkrat omenili raznolikosti med posameznimi vgradnimi omarami.

Literatura

[1] (2014) CAD. Dostopno na:
http://www.techterms.com/definition/cad

[2] (2014) Computer-aided design. Dostopno na:
http://en.wikipedia.org/wiki/Computer-aided_design

[3] (2014) CAD / Computer-Aided Design. Dostopno na:
http://www.plm.automation.siemens.com/en_us/plm/cad.shtml

[4] (2014) CSV Files. Dostopno na:
http://www.csvreader.com/csv_format.php

[5] (2013) SolidWorks Fact Sheet. Dostopno na:
https://www.solidworks.com/sw/docs/Corp_FactSheet_2013Q2.pdf

[6] Jami J. Shah, *Parametric and Feature-Based CAD/CAM: Concepts, Techniques, and Applications*, John Wiley & Sons, 1995

[7] O.W. Salomons, F.J.A.M. van Houten and H.J.J. Kals, "Review of research in feature-based design", *Journal of Manufacturing Systems*, št. 2, zv. 12, str. 113–132, 2003

[8] (2014) Approach to Modeling. Dostopno na:
http://help.solidworks.com/2014/English/SolidWorks/acadhelp/c_Approach_to_Modeling.htm?id=ecc2097807c740d8b860a2b7a887de84#Pg0

[9] Alex Ruiz, *SolidWorks 2010: No Experience Required*, John Wiley & Sons, 2010

[10] SolidWorks Corporation, *SolidWorks Essentials: Training*, Dassault Systemes SolidWorks Corporation, 2012

[11] Matt Lombard, *Solidworks 2013 Bible*, John Wiley & Sons, 2013

[12] Luke Malpass, *SolidWorks API Series 1: Advanced Product Development*, AngelSix, 2014

[13] (2014) SolidWorks C# Add-in Template. Dostopno na: http://help.solidworks.com/2014/English/api/sldworksapiprogguide/Overview/Using_SolidWorks_C__Add-In_Wizard_to_Create_C__Add-In.htm?id=7073034e1c0e4490804563a232186867#Pg0

[14] (2007) OpenSceneGraph: Introduction. Dostopno na: http://trac.openscenegraph.org/projects/osg/wiki/About/Introduction